INTRODUCTION TO NATURAL LANGUAGE PROCESSING

Reinforcement Learning & Alignment

Victor Zhong

Introduction to Natural Language Processing

The Limits of Instruction Tuning

Recap: SFT (Instruction Tuning) teaches a model to follow instructions by imitating human examples.

Problem 1

Human annotators make mistakes.

Problem 2

It is easier to judge a good answer than to write one.

Problem 3

Imitation Learning has a ceiling; the model can only be as good as the average annotator.

The "Alignment" Goal

We want models to be:

Helpful

Follow the user's intent.

Honest

Avoid hallucinations and deception.

Harmless

Avoid generating toxicity, hate speech, or dangerous instructions.

SFT gets us mostly there, but RLHF is the fine-polishing step.

Preference vs. Generation

Insight

Humans are better **critics** than **creators**.

- It is hard to write a perfect poem.
- It is easy to look at two poems and say "Poem A is better than Poem B."

Idea

Can we train a model using this easier signal (Preferences) instead of expensive demonstrations?

What is Reinforcement Learning?

Definition: RL is learning by trial and error. An agent takes actions in an environment to maximize a cumulative reward.

Supervised Learning

"Here is the correct answer."

(Teacher)

Reinforcement Learning

"That was a good/bad attempt."

(Critic)

Key RL Concepts (1)

Agent

The learner.

(The LLM)

Environment

Everything outside the agent.

(The User / The Chat Interface)

State (s)

The current situation.

(The conversation history / Prompt)

Action (a)

What the agent does.

(The generated token or response)

Key RL Concepts (2)

Policy (π)

The strategy. A function mapping States to Actions.

$$\pi (a | s) = P (\text{next token} | \text{context})$$

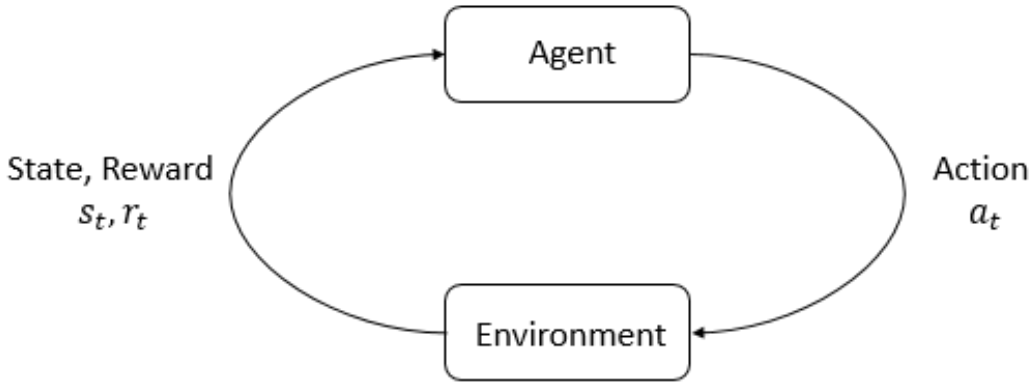
Reward (r)

A scalar feedback signal.

- **High reward (+10):** Good response.
- **Low reward (-100):** Toxic response.

The RL Loop

- Agent observes **State** s_t.
- Agent executes **Action** a_t based on Policy π.
- Environment returns **Reward** r_t and next state s_{t+1}.
- Agent updates Policy to maximize future rewards.



The cyclical interaction between Agent and Environment.

$$J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} r_t\right]$$

Markov Decision Process (MDP)

Formal framework for RL.

Markov Property

The future depends only on the current state, not the history.

Note: In LLMs, the "State" is the entire context window, so it implicitly includes history.

Trajectory: A sequence of states and actions:

$$\tau = (s_0 , a_0 , r_0 , s_1 , a_1 , r_1 \ \dots \ )$$

Value Functions

Return (G_t)

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

The total accumulated reward from time t .

Value Function ($V(s)$)

$$V^{\pi}(s) = \mathbb{E}_{\pi}[G_t \mid S_t = s]$$

How good is it to be in state s ? (Expected future reward).

Q-Function ($Q(s, a)$)

$$Q^{\pi}(s, a) = \mathbb{E}_{\pi}[G_t \mid S_t = s, A_t = a]$$

How good is it to take action a in state s ?

Policy Gradient Methods

Algorithm	Ψt (The "Weight")	Why use it?
REINFORCE	G_t	Simple, unbiased, but high variance.
Q-Actor-Critic	Q(s, a)	Lower variance, requires learning a Q-function.
Advantage AC	Q(s, a) - V(s)	Best of both worlds; tells you if an action was *better than average*.

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)\Psi_t\right]$$

We want to increase the probability of actions that lead to high rewards.

https://lilianweng.github.io/posts/2018-04-08-policy-gradient/

Where does the Reward come from?

In Games (Chess/Go)

The environment gives the reward (Win/Loss).

In Chatbots

There is no automatic "Win" button.

Solution: We must learn a Reward Function from human data.

Collecting Preference Data

We give annotators a prompt and two model responses.

Prompt: "Explain quantum computing."

Response A: "It uses qubits..."
Response B: "I don't know."

Label: A > B

The Bradley-Terry Model

How do we turn "A > B" into a number?

We assume there exists a latent "score" (reward) $r(x, y)$ for every prompt x and response y.

Probability that A is preferred to B

$$P(A \succ B) = \sigma(r_A - r_B)$$

σ is the Sigmoid function.

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Training the Reward Model (RM)

- **Architecture of r:** Take a pretrained LLM, replace the final layer with a scalar head.
- **Input:** Prompt + Response.
- **Output:** Score (Scalar).

Loss Function

Minimize the error in predicting human preferences.

$$\mathcal{L}(\phi) = -\frac{1}{K} \sum_{(x, y_w, y_l) \sim D} \log\left(\sigma(r_\phi(x, y_w) - r_\phi(x, y_l))\right)$$

Reward Model Challenges

Ambiguity

Annotators disagree (subjectivity).

Reward Hacking

The agent finds a way to get a high score without actually being helpful (e.g., being extremely verbose).

Proxy Gap

The Reward Model is just a proxy for human preference, not the preference itself.

The RLHF Pipeline

The standard method (Ouyang et al., 2022) has three steps:

1

SFT

Train a supervised baseline.

2

RM

Train a Reward Model on comparison data.

3

PPO

Use Proximal Policy Optimization to fine-tune the LLM against the Reward Model.

Step 3 - PPO (High Level)

$$R(x, y) = r_\phi(x, y) - \beta \log \left(\frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)} \right)$$

Policy (π_θ)

The LLM we want to align.

Environment

The prompt dataset.

Reward Signal

The output of our trained Reward Model.

Goal: Maximize Reward while not changing too much from the SFT (ref) model.

The KL Divergence Penalty

$$R(x,y) = \underbrace{r_\phi(x,y)}_{\text{Reward Model}} - \beta \underbrace{\mathbb{D}_{KL}(\pi_\theta(y|x) \parallel \pi_{\text{ref}}(y|x))}_{\text{KL Penalty}} \qquad \mathbb{D}_{KL} = \log\left(\frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}\right)$$

If we just maximize reward, the model will output gibberish that tricks the Reward Model.

Solution: Add a penalty if the model drifts too far from the original SFT model (π_{ref}).

Why KL Penalty Matters

β (Beta): Controls the strength of the constraint.

$$R(x, y) = \underbrace{r_\phi(x, y)}_{\text{Reward Model}} - \beta \underbrace{\mathbb{D}_{KL}(\pi_\theta(y|x) \parallel \pi_{\text{ref}}(y|x))}_{\text{KL Penalty}}$$

High β

Model stays very close to SFT (Safe, but less improvement).

Low β

Model optimizes heavily for reward (Risks mode collapse or hacking).

It ensures the language remains fluent and diverse.

Mode Collapse in RLHF

If β is too low in DPO/PPO, the model loses entropy.

$$R(x, y) = \underbrace{r_\phi(x, y)}_{\text{Reward Model}} - \beta \underbrace{\mathbb{D}_{KL}(\pi_\theta(y|x) \parallel \pi_{\text{ref}}(y|x))}_{\text{KL Penalty}}$$

- It starts outputting the exact same "safe" answer for every prompt.
- *"As an AI language model, I cannot..."*

Maintaining diversity is the hardest part of alignment.

PPO Algorithm Details

1. Rollout

Generate a response using the current LLM.

2. Evaluation

Score it with the Reward Model.

3. Update

Calculate gradients to increase the likelihood of high-scoring tokens.

4. Constraint

PPO clips updates to prevent destructive large jumps in policy.

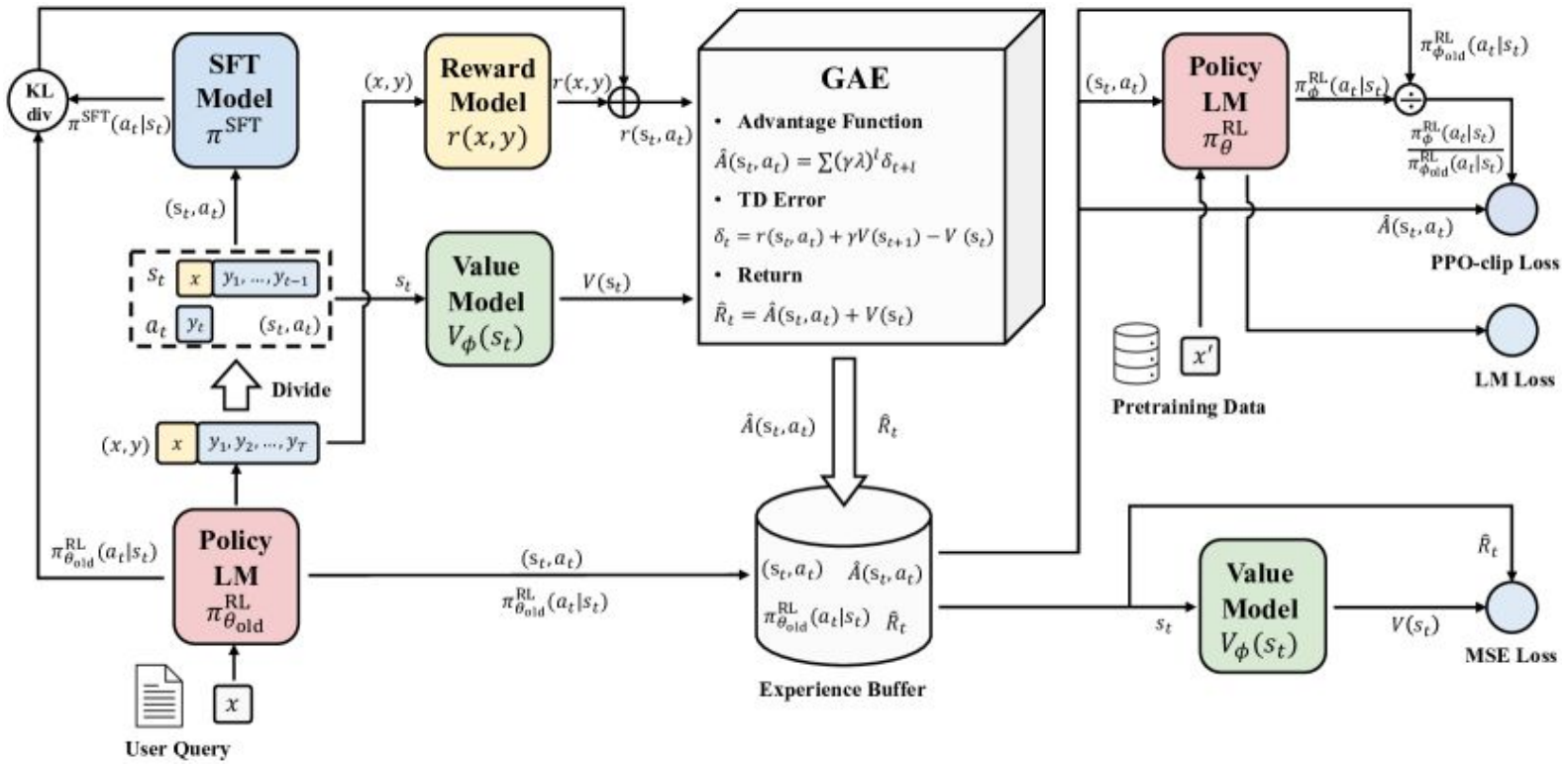
PPO clipping prevents the model from forgetting everything it learned in the SFT stage by taking too big a step.

The Problem with PPO

PPO is complex and unstable.

It requires loading **4 models** into memory simultaneously:

- The Policy (Actor)
- The Reference Model
- The Reward Model
- The Value Model (Critic)



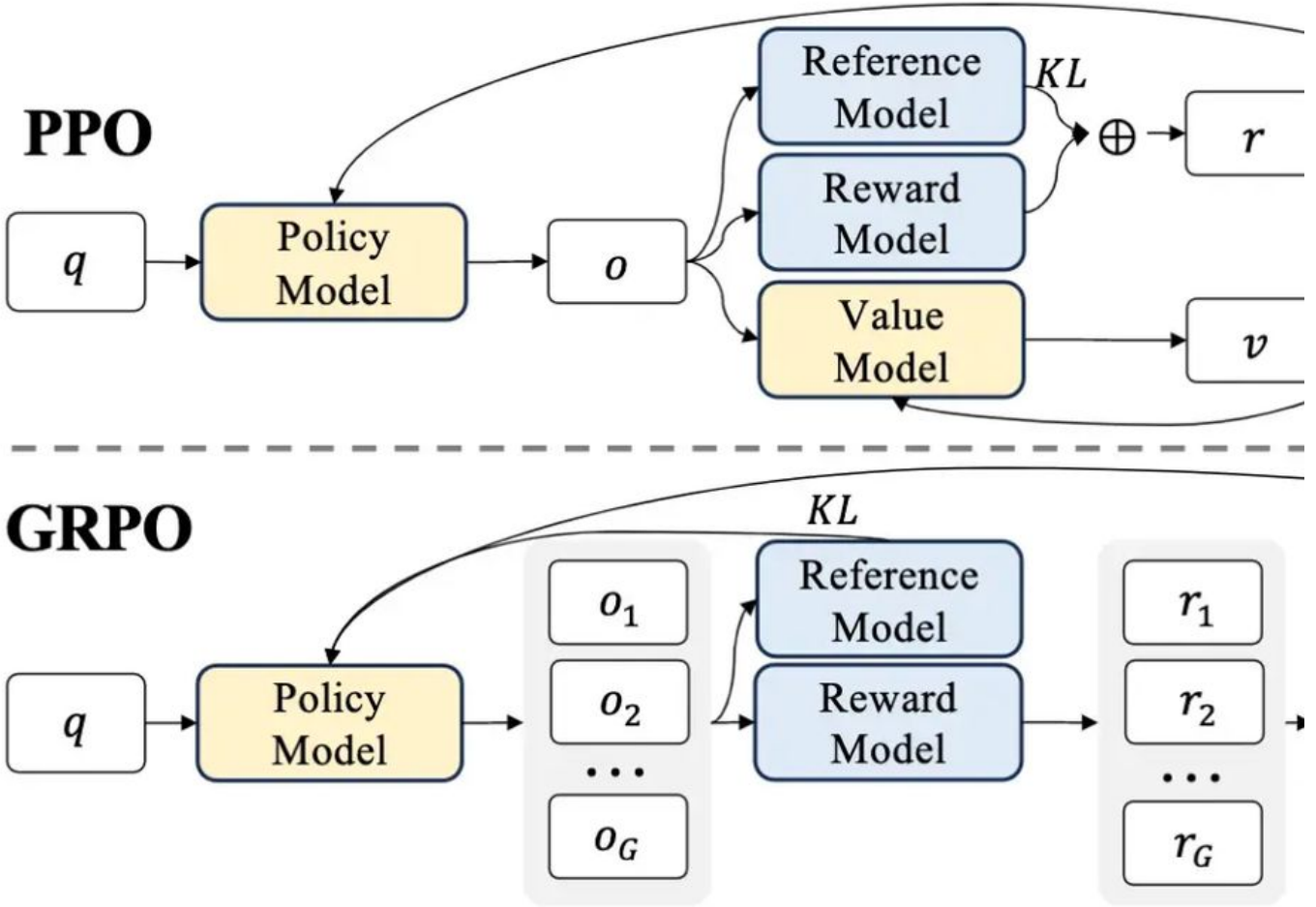
https://arxiv.org/abs/2307.04964

Group Relative Policy Optimization (GRPO)

GRPO eliminates value critic model

Instead: use group statistics

- generates N responses for same prompt
- calcucates reward for each and uses group mean and STD to derive relative advantage



https://ghost.oxen.ai/why-grpo-is-important-and-how-it-works/

DPO: A Simpler Approach

$$\mathcal{L}_{DPO}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x,y_w,y_l)\sim D}\left[\log \sigma\left(\beta \log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right)\right]$$

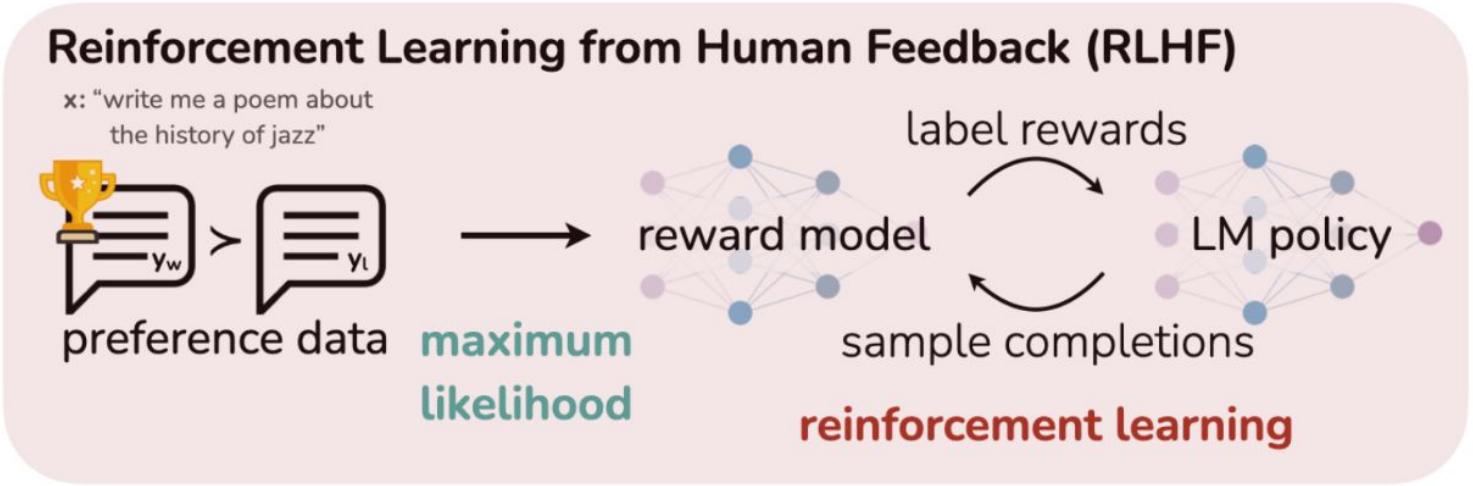
Direct Preference Optimization (DPO) (Rafailov et al., 2023).

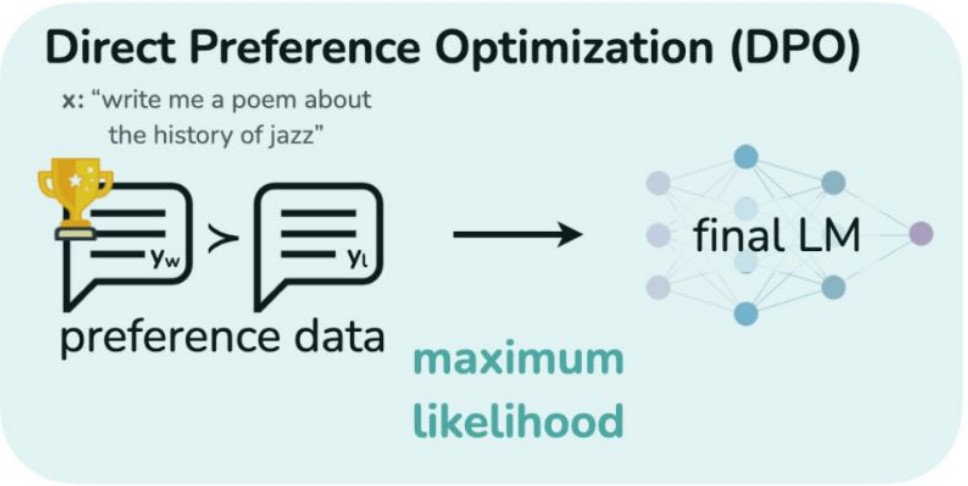
Key Insight

We don't need a separate Reward Model.
The Language Model itself can implicitly define the reward.
We can solve the RLHF mathematical problem directly without an RL loop.

The DPO Mathematical Trick

$$\mathcal{L}_{DPO}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x,y_w,y_l)\sim D}\left[\log \sigma\left(\beta \log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right)\right]$$





The DPO Loss Function

$$\mathcal{L}_{DPO}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x,y_w,y_l)\sim D}\left[\log \sigma\left(\beta \log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right)\right]$$

Intuition: Increase the likelihood of the winner (y_w) and decrease the likelihood of the loser (y_l), weighted by how much the reference model liked them.

- reward replace by $\beta \log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}$
- policy makes “winner” more likely than reference: log ratio goes up
- policy makes “loser” less likely than reference: log ratio also goes up
- gradient increases likelihood of winner and decreases likelihood of loser, weighted by how much model already prefers winner over loser

DPO vs. GRPO vs. PPO

Feature	PPO (RLHF)	GRPO	DPO
Complexity	High (RL Loop)	Medium	Low (Supervised-like)
Models in Memory	4	3	2
Stability	Unstable (Hyperparams)	Stable	Stable
Performance	Excellent	Comparable	Comparable

DPO Workflow

1. SFT

Train baseline.

2. Data

Collect pairs (x , y_w , y_l).

3. Training

Run DPO directly on this data.

- No separate Reward Model training step.
- No sampling/rollouts during training (much faster).

KTO (Kahneman-Tversky Optimization)

$$\mathcal{L}_{KTO} = \mathbb{E}_{x,y}\left[w(y) \cdot \text{loss}\left(\sigma\left(\beta \log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)} - z_{ref}\right)\right)\right]$$

DPO requires pairs of preferences (A > B).

Sometimes we only have "Thumbs Up / Thumbs Down" data (unpaired).

KTO

- An alignment method that works with unpaired binary feedback.
- Easier to collect data for KTO (e.g., users clicking "Like" on ChatGPT).
- zref is a “reference point”, typically expected log-ratio across all examples.
- w(y): accounts for class imbalance (more thumbs up than thumbs down

Without Annotators: Constitutional AI

Can we align models without humans?

Idea: Use an LLM to critique and revise its own outputs based on a "Constitution" (a set of principles).

1. Generate

Generate harmful response.

2. Critique

Ask LLM to critique it based on principles.

3. Rewrite

Ask LLM to rewrite it.

4. Train on the rewritten data (RLAIF - RL from AI Feedback).

Without Annotators: Rejection Sampling

A simple alternative to PPO/DPO.

$$\mathcal{L}_{RS}(\theta) = -\mathbb{E}_{x\sim D}[\log \pi_\theta(y^*|x)]$$

Algorithm

- Generate N responses for a prompt.
- Score all N with a Reward Model.
- Keep the best one.
- Fine-tune the model on the best responses (SFT).

Popular in Llama 2/3 training ("Iterative SFT").

The Alignment Tax

Does making a model safer make it dumber?

Alignment Tax

The potential drop in performance on objective tasks (like coding or math) when tuning for helpfulness/safety.

Early RLHF models suffered from this; modern techniques (like DPO) minimize it.

Sycophancy

RLHF models tend to agree with the user, even when the user is wrong.

Why? Humans prefer answers that validate their beliefs. The Reward Model learns this bias.

Example

User: "I think the earth is flat."
Model: "That's an interesting perspective..." (Instead of correcting).

Hackability / Jailbreaks

Alignment is not a guarantee.

Jailbreaks

Clever prompts (e.g., "Do anything now (DAN)") can bypass safety filters.

Adversarial Attacks

Appending specific random character strings to force the model to output harmful content.

Scalable Oversight

Problem

How do we align models that are smarter than us?

If an LLM generates a complex mathematical proof or code, a human annotator might not be able to judge if it's correct.

Research

We need methods where weak supervisors (humans) can control strong agents (superintelligent AI).

Summary (1/2)

- **Alignment:** Adjusting model behavior to match human intent (Helpful, Honest, Harmless).
- **Preference Learning:** Using comparison data (A > B) is more scalable than writing demonstrations.
- **Reward Models:** Neural networks trained to act as the critic.

Summary (2/2)

- **PPO:** The classic RL approach. Optimizes policy against a reward model with KL penalties.
- **GRPO:** New memory-efficient RL that uses group statistics instead of a critic model.
- **DPO:** The modern approach. Optimizes preferences directly without an explicit reward model.
- **Future:** Moving toward AI feedback (Constitutional AI) to scale oversight.

Fine-tuning for safety and subtle preferences.

The Full Training Recipe

1. Pretraining

Raw knowledge (Web text).

2. SFT

Formatting and instruction following (Q&A pairs).

3. Rejection Sampling

Filtering for best responses.

4. DPO/GRPO/PPO

Adjusting model behaviour based on preferences on live data.

Final Thoughts

- Reinforcement Learning is the engine of modern AI agency.
- It bridges the gap between predicting the world and acting in it.
- Alignment is an unsolved research problem—we are just scratching the surface.

Appendix: RL Terminology

On-Policy (PPO)

The agent learns from data it just generated.

Off-Policy (DQN)

The agent learns from old data stored in a replay buffer.

Model-Free

We don't learn a physics simulator of the world; we just learn Value/Policy directly.

Appendix: Elo Ratings

We can rank models using Elo ratings. Once match occurs (preference expressed), update ratings based on how likely actual outcome was

$$E_A = \frac{1}{1 + 10^{(R_B - R_A)/400}}$$

$$R'_A = R_A + K(S_A - E_A)$$

E_A: probability that A will beat B.

S_A: actual outcome

K: parameter deciding how fast ratings change

Chatbot Arena

Uses Elo rating system (from Chess).
Based on blind pairwise comparisons by humans.

Current SOTA is dominated by models heavily tuned with RLHF/DPO.

Key Papers (1)

- **PPO:** Schulman et al. (2017). "Proximal Policy Optimization Algorithms".
- **InstructGPT (RLHF):** Ouyang et al. (2022). "Training language models to follow instructions with human feedback".
- **Deep Reinforcement Learning:** Mnih et al. (2015). "Human-level control through deep reinforcement learning".

Key Papers (2)

- **DPO:** Rafailov et al. (2023). "Direct Preference Optimization: Your Language Model is Secretly a Reward Model".
- **GRPO:** Shao et al. (2024). "DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models
- **Constitutional AI:** Bai et al. (2022). "Constitutional AI: Harmlessness from AI Feedback".
- **Lilian Weng's Blog:** "A (Long) Peek into Reinforcement Learning" (2018).